Karsenty Eva –[redacted] Robbes-Lasry Raphael -[redacted] Sela Amitai –[redacted]

Project Report

Introduction :

link to github:
https://github.com/Amitaisela/Project_lab

For this project we choose to develop a use for everyday users. We choose to build an AI-driven solution that recommends companies based on an individual's linkedin profile. In the current market, identifying a good company that fits you can be overwhelming, often resulting in wasted time and resources for both the company and the person. Our approach aims to resolve this issue, by using the user's attributes, to find the three most suitable companies. Using AI, we offer a decision-making tool that enables users to make informed selections.
In the appendix there is the updated proposal of the project.

Data Collection and Integration:

link to data:
https://technionmail-my.sharepoint.com/:f:/g/personal/amitaisela_campus_technion_ac_il/EomP33IaaI1Mm7ivkRZ8Ni8BGgdeTAU8CrxwaDdU4AC0fQ?e=qvBu48

The given data:

In order to build the best model, we choose to save only relevant features that could help our "matching model". Therefore, from our original linkedIn profiles dataset, we choose to use only certain features, as shown in the appendix:
We chose these features as they offer insights on an individual's experience, interests, training, location, skills and more.
These characteristics are crucial for accurately identifying and matching individuals with the most suitable companies. By considering these key factors, we can enhance the precision of our recommendations and ensure a better fit between individuals and companies. Furthermore, we opted to exclude profiles lacking a current **and** previous company, as well as individuals associated with companies appearing fewer than four times in our dataset. This curation process ensures the quality and relevance of the data, refining our analysis to focus on more robust and representative profiles. After this filtration, we got ~230K profiles. In the last dataframe that we had, with the ~230K profiles, their is no column with a null.

The scraping data:

For scraping, we collected additional linkedIn profiles from Israel (which is a different country from our original dataset). To accomplish this task, we used the Python library **selenium** and the bright-data framework. Using selenium, we took Israeli companies listed on LinkedIn, found their ID's, iterated through each company, going to a **very specific url** that displays employees from these companies who are from Israel, and extracted the URLs of their profiles. All collected URLs were cleaned and compiled into a CSV file, which was then uploaded to Bright Data web IDE linkedin profile scraper. The scraper produced a CSV file containing the newly collected data, comprising profiles totaling ~1800 rows and 26 columns. In this additional data, we also choose to keep only relevant features that were not exactly the same as in our original dataset, but they follow the same logic. The exact features are in the appendix.

Our additional data was integrated in our solution by concating it into the original data. This enriched understanding leads to more informed recommendations, that are also based on the place the user is in.

As explained in the instructions and in the reception time, our $\alpha(group)$ = 1.845 since we have 1845 rows and the $\alpha(group)$ is a function of multiplication of 1000.

A simple breakdown of the distribution of the company sizes is in the appendix.

Data Analysis:

First, we began by transforming our data features into vectors to facilitate calculations. To accomplish this, we utilized the "small_bert_L2_128" pre-trained embedding from BERT.

We combined all the features into a unified string format and subsequently embedded them. Given our intended usage of most columns in the feature set, more statistical analysis was unnecessary - we choose a column to put in the features if it describes an aspect of a person, as we said in the introduction.

Another analysis involved evaluating the prevalence of less significant labels:

Labels appearing 1 time(s): 264,888
Labels appearing 2 time(s): 49,033
Labels appearing 3 time(s): 19,698
Labels appearing 4 time(s): 10,628
Labels appearing 5 time(s): 6,592

Following this assessment, we removed rows containing labels that appeared fewer than five times to prevent potential interference with our model's performance.

We tried to make useful visualizations on the given dataset, but we couldn't find a lot. we made a number of employees in company distribution, like in the scraping, as shown in the appendix.

AI and ML Methodologies:

We used 2 models - an ML model - KNN (K-nearest-neighbors) and a NN model.

- For only the given dataset, after iterating through K's, we have found that K=19 is the best k value for accuracy, but unsurprisingly, got us only an accuracy of ~0.2.
- For both the given dataset and the scraping set, after iterating through K's, we have found that K=27 is the best k value for accuracy, but with a similar accuracy of ~0.2.

When we applied KNN on **only** our scraped data, we found that the best k-value is 19, with an accuracy of more than double the previous value - ~0.55. We suspect we got this high accuracy because of the class balance of the scraped dataset, while the given and filtered data classes are very much imbalanced.
KNN provides a simple and interpretable approach to recommending companies to users, yet not really reliable on imperfect dataset such as we have.

To build the model using neural networks and their types. The first thing we did was to implement and test an RNN and a LSTM model, but both of the results we obtained were not conclusive, without any of them converging, with accuracy and loss all over the place, with an enormous running time.

Finally, we chose to implement by ourselves a NN (Neural Network) that gave us our best results.
This model is structured with 2 layers. It begins with an input layer followed by a fully connected layer ,512, applying batch normalization, and a hyperbolic tangent activation function. Dropout regularization is then applied to mitigate overfitting, the data flows through another fully connected layer, 512, with similar batch normalization and activation, followed by dropout. Finally, the output layer produces the model's prediction. This architecture is designed to balance complexity and regularization. We run it on an adam optimizer, batch of 64, with only 75 epochs. That was because of our time limitation - we believe that with more powerful machines, and with more times, higher epochs and batch sizes would yield a higher result.
We choose 512, 512 layer size because of some factors:

1) a higher hidden layer size didn't change too much the accuracy of the model on the same number of epochs.
2) We tried to add more hidden layers - that didn't help the model, and some variations even hurt it.
3) In the end, trying to use a truly huge and complex model, based on our limited experience, could take somewhere from 30 minutes to 6 hours **per epoch** - We simply don't have this kind of time.

We run the NN 3 times - **one** for just our filtered data, without the scraping data included, and **one** with the scraping data included.
We will discuss the results in the results section, but we will mention here that we tried to also run it only on the scraped data, and got an immediate of 0.99 accuracy, and loss of almost 0 - we didn't

believe that this model can really be used, because it almost certainly overfit on the small dataset the scraped data we provided.

Evaluation and Results:

For each of our models used, we implement the evaluating process in the following way:
Initially, we split our dataset into training, and testing sets to evaluate model performance accurately. Also, to get the most of our model, we made sure that all of the labels are in both the train and test datasets.
We checked precision, accuracy, f1, recall in the KNN, to quantify the system's performance. In the NN we just choose accuracy. Fine-tuning hyperparameters and testing the optimized model completed the evaluation process, ensuring the system's suitability for real-world deployment.
For KNN, we received an accuracy of 0.2, both on the regular data and the regular + scraped data - and 0.55 for the only-scraped data, which was not very conclusive. Hence, to enhance accuracy, we explored alternative models.
As we mentioned before, with RNN and LSTM, our accuracy and loss didn't stabilize to a score, rendering the results inconclusive. Consequently, we experimented with our NN model, yielding an accuracy of ~0.4 for both with and without the scraping data, when recommending the top 3 companies corresponding to the user profile. Therefore, we can conclude that our more performant model was our NN model. It is important to note that the model's accuracy were definitely still rising with the epochs - with a higher number of epochs, we believe that we would have gotten a higher accuracy on all models.
It is not very surprising that adding the scraped dataset didn't add too much for the accuracy, so in order to check our how well the model does on the scraped data, we took the test dataset of this model, extracted only the **scraped data** from it, and checked the model accuracy only on it.
In the test portion on the scraped data, we had 168 unique labels. when we run the model of the test portion of the scraped data alone, we got:
Test Loss: **4.1974**, Top-3 Test Accuracy: **0.4949 -** better than the model got on the total average! We believe that this means that our model is able to distinguish between the Israeli and Us profiles pretty well.

Limitations and Reflection:

During our project, we encountered different challenges and constraints.

1. We had challenges in training and experimenting with complex models like deep neural networks, because of the running time of each model we built. This constraint affected the scalability and efficiency of our model training process.
2. We encountered challenges in achieving convergence with recurrent neural network (RNN) and long short-term memory (LSTM) models. This finding emphasizes the importance of selecting appropriate architectures for the problem domain.

3. The databricks notebook had crushed multiple times on us, making us run everything again

The worst limit we had is definitely the **slowness** and the lack of **more powerful machines** to run our model on. Those limits certainly held us back from finding the best model we can.

Conclusion:

KNN:

We dont believe in the model's ability to find the proper similarity between our users, due to its high complexity. An idea we got in order to solve this is to form the embedding, cast it into a much higher dimension, in order to get more complexity for the KNN to rely on - but this leads us straight to the Curse of Dimensionality, which can’t be ignored.

NN:

Our NN did pretty well with and without the scraped data - a 0.4 accuracy score is good, because it keeps the current company of the user in mind, while letting him “branch” into other companies, and a 0.4949 on only the scraped data test portion is well beyond what we believed our model could do.

In order to make the performance better, we would need a stronger machine to make our testing and tuning time-efficient. Also believe that using 2 layers, and making the hidden layer dimension higher, with a higher number of epochs, could give us better results.

Exploring more avenues in the future, we contemplated adding company similarity metrics into our feature set. Rather than focusing on accuracy regarding the top three predicted labels, we suggest evaluating model performance based on the similarity between these labels and the truth label.

APPENDIX:

Introduction - features:

original Dataset relevant features:

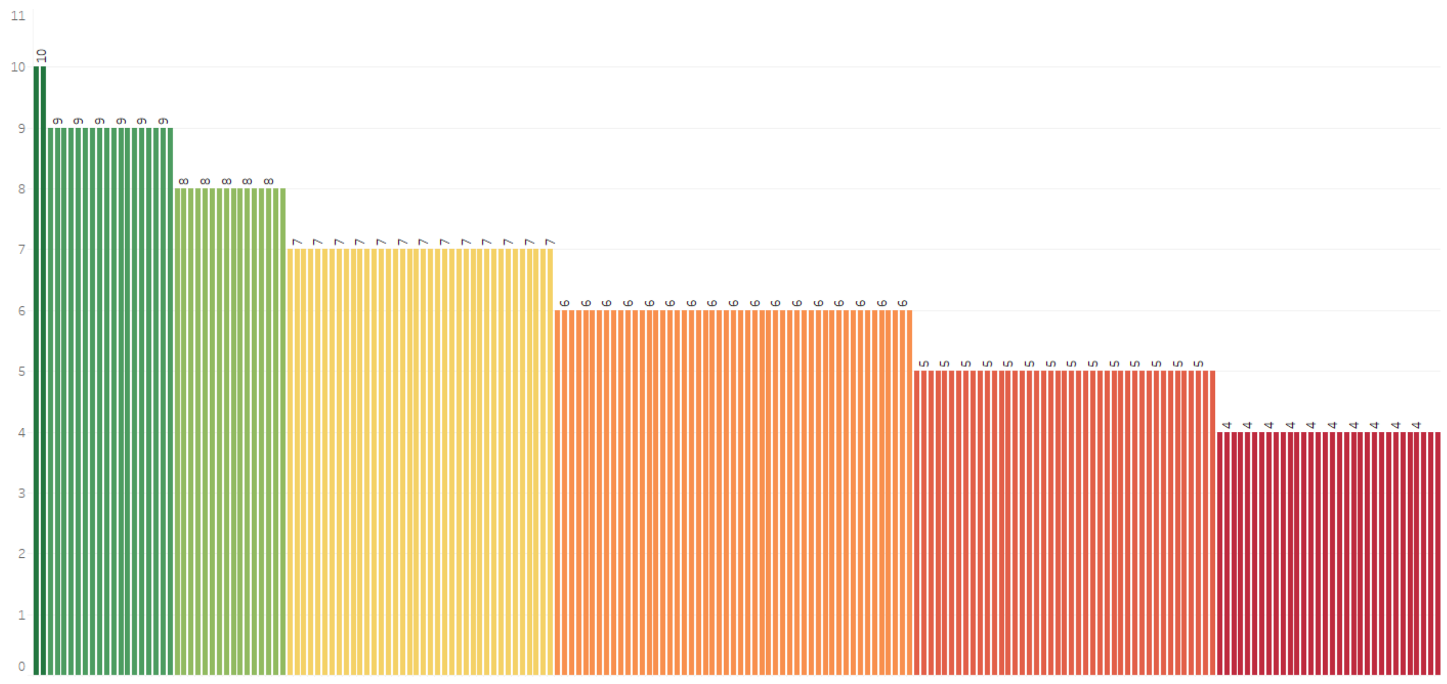
about, **certifications**, **city**, **current company industry**, **current company title**, **current company id**, **current company name**, **degree, field**, **experience company**, **experience description**, **experience duration**, **positions title**, **experience title**, **id**, **language level**, **languages**, **position**, **recommendations count.**

Scraped dataset relevant features:

region, **position**, **education details**, **recommendation count**, **current company name**, **current company id**, **projects info**, **patents info**.

Data Collection and Integration:

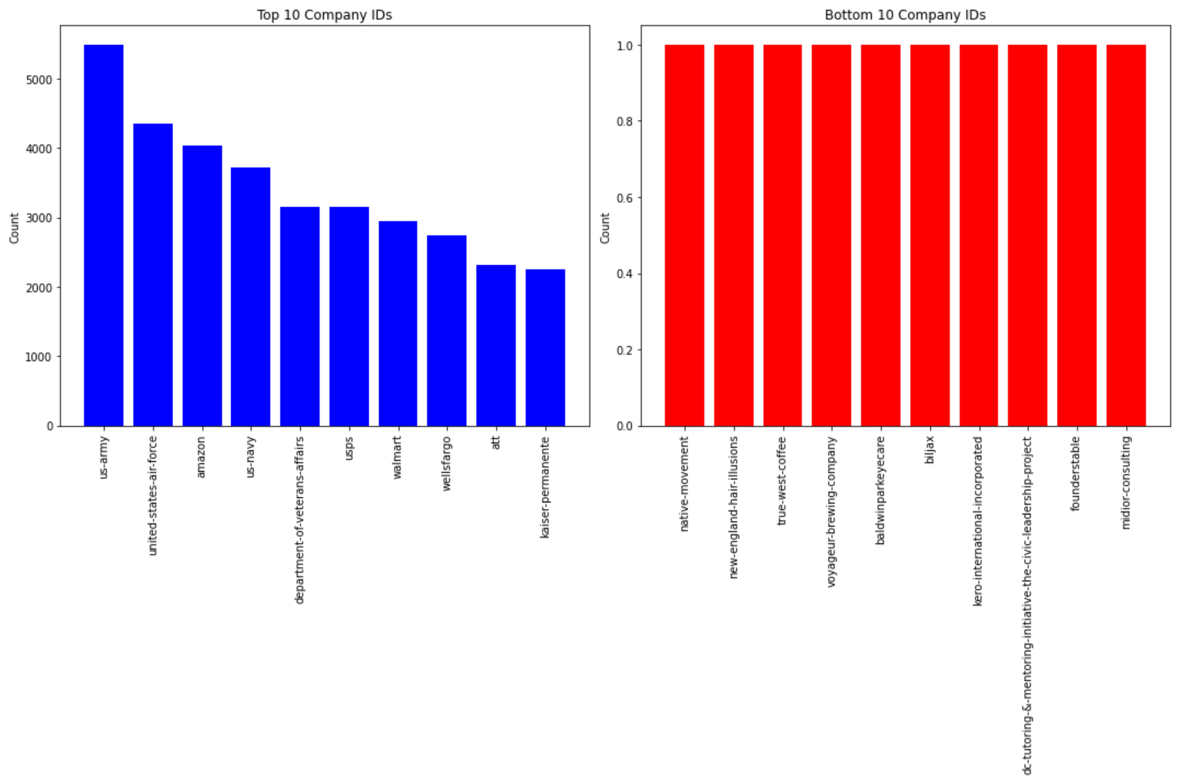
number of employees in company distribution:



under the viz, there were supposed to be company names, but it got so messy we did not show it (it's not relevant anyways).

Data analysis:

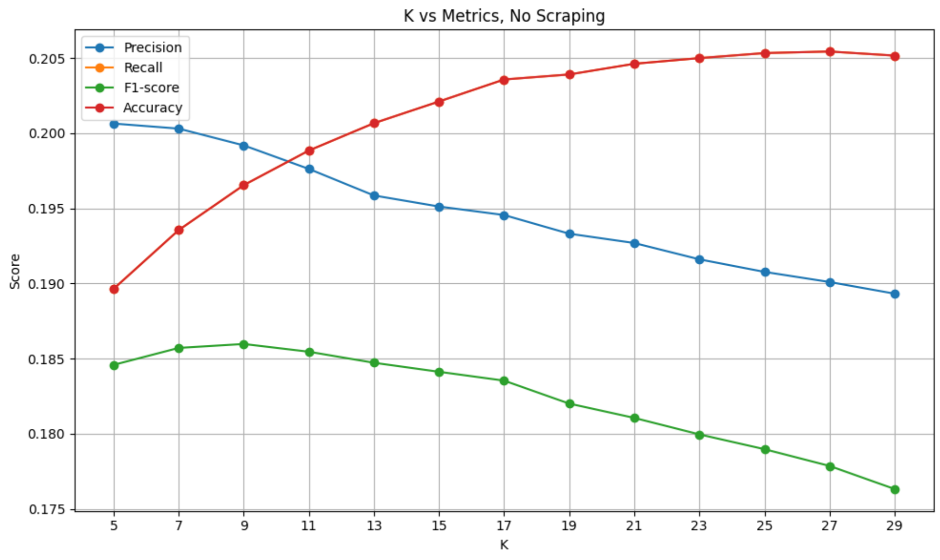
Number of employees in company distribution.



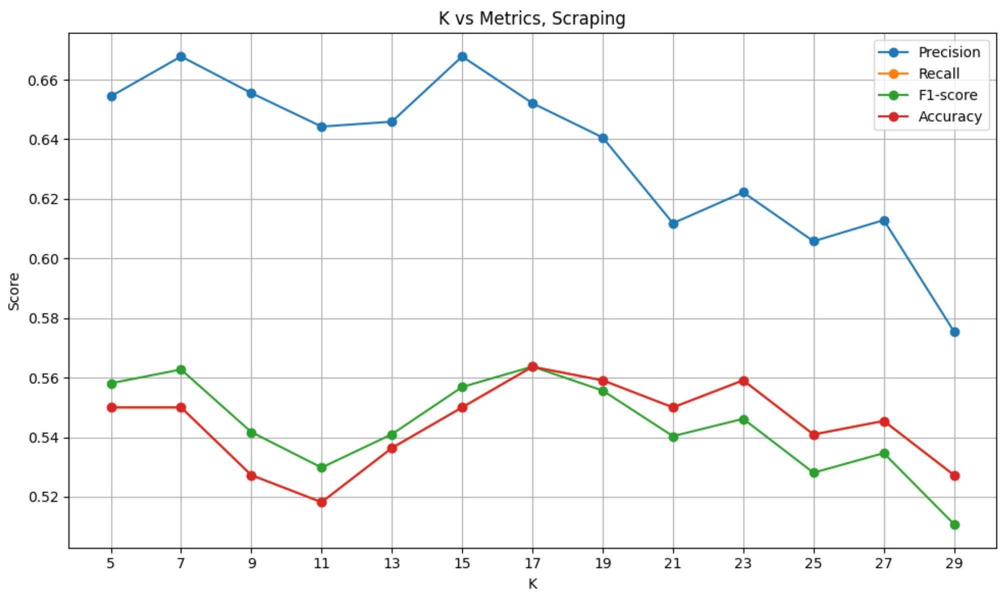
Evaluation and Results - images:

KNN:

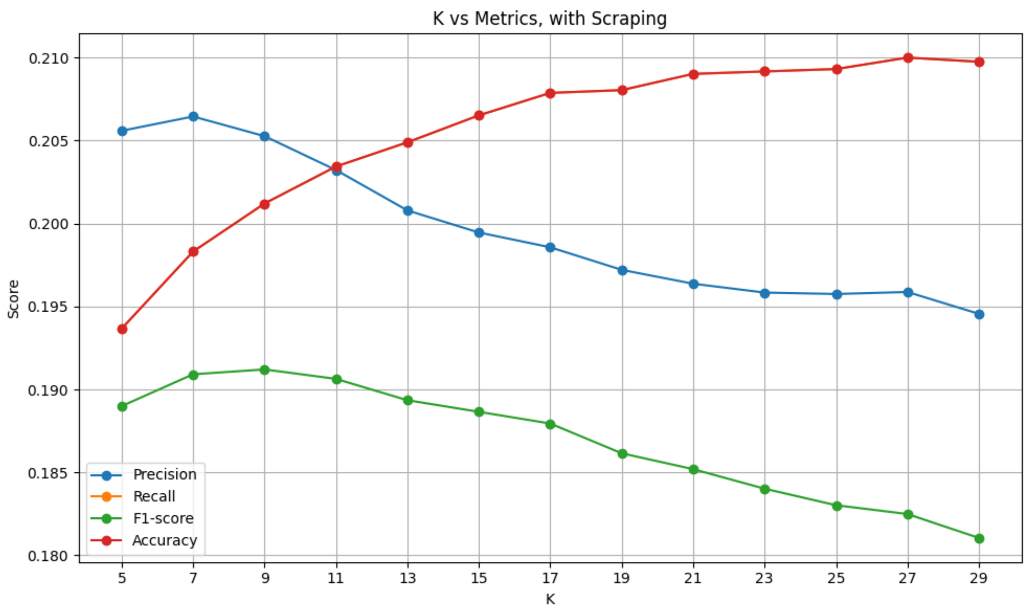
The result of the KNN without the scraping data:



The result of the KNN of only the scraping data:

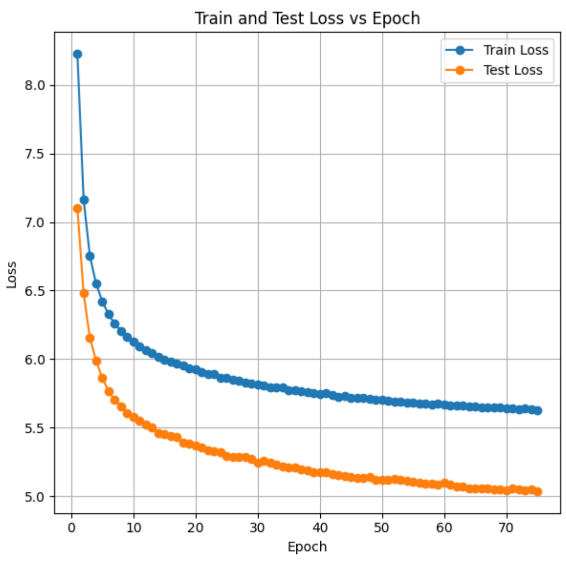


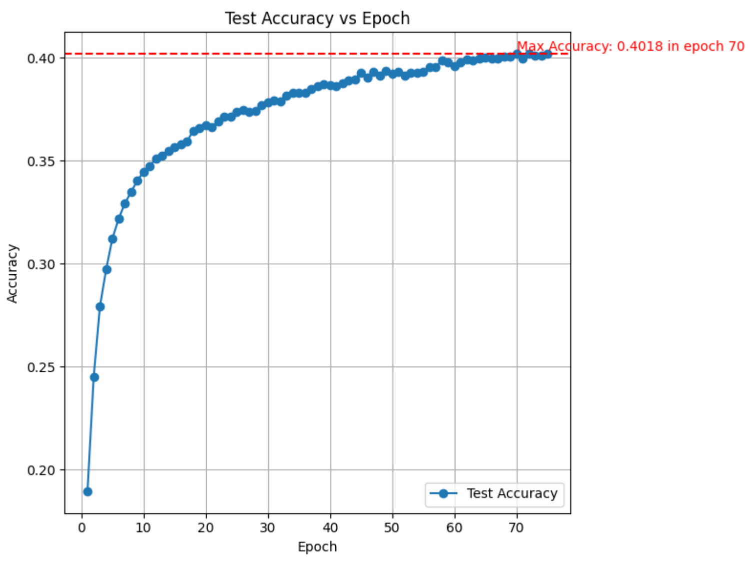
The result of the KNN of with both the scraping data and the regular data:



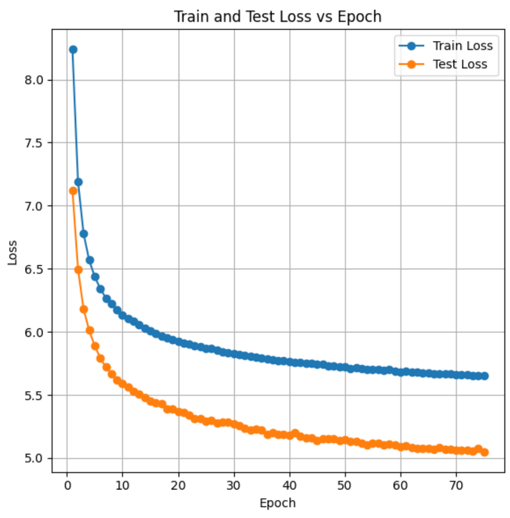
NN:

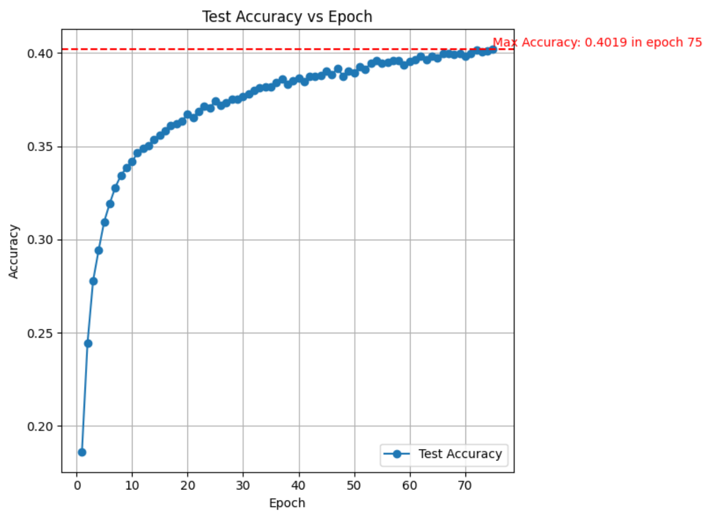
Without the scraping data integration:





With the scraping data integration:





Project Proposal

Introduction and Chosen Path:

Our project aims to develop an AI-driven solution tailored to the needs of individual job seekers. Instead of focusing solely on companies seeking the best candidate matches, we are switching our focus to another project direction job seekers to identify the companies that best align with their skills, experiences, and preferences. This user-centric approach allows us to provide to the broader market of job seekers and provide personalized recommendations tailored to their unique profiles.

Research Question:

How can we leverage AI algorithms to analyze individual job seeker profiles and assist them in identifying the companies that best match their skills and preferences?

Ways of solving:

- For the beginning we will start trying to do KNN (K-nearest-neighborhoods) and find some statistics.
- After that we will do harder work to try implementing a Neural Network (SimpleNN/ComplexNN/RNN/LSTM) that will fit our data and could "learn" our data the best, so we could receive some insights from him.

Benefits of External Solutions for users use:

1. **Personalized Job Recommendations:**
    - Our AI product will provide job seekers with personalized recommendations based on their skills, experiences, and preferences. By analyzing individual profiles, job seekers can identify the companies that offer the best fit for their career aspirations.
2. **Empowering Job Seekers:**
    - Our solution empowers job seekers to take control of their job search process. By proactively identifying potential employers that align with their career's millstones, job seekers can optimize their job search efforts and increase their chances of finding the right company for them.

Relevant Data from General Databases:

People Dataset:

We will continue to extract relevant information from job seeker profiles, including experiences, education, and skills. This dataset is the profiles dataset we got from LinkedIn.

Additional Data Collection:

We will focus on building and training our model using data from the USA initially. However, we recognize the importance of evaluating its effectiveness across different regions. Therefore, we will collect additional data from Israel, to assess the model's performance in predicting company matches for job seekers in diverse geographical contexts.

For the $\alpha(group)$ we estimate to receive 1.5 < $\alpha(group)$ < 2. As explained that $\alpha(group)$ is a function of multiply by 1000.

Project End Goals:

Our final goal is to develop a user-centric AI tool that empowers job seekers to identify the companies that best match their skills and preferences. By leveraging AI algorithms and data analysis like we explain above, we aim to provide job seekers with personalized information, a way that enhance their job search experience and could lead to a more successful career outcomes. Through continuous evaluation and improvement, we strive to ensure the effectiveness and reliability of our solution across diverse user profiles and countries.